de Jong et al. Journal of Cheminformatics 2013, 5:25
http://www.jcheminf.com/content/5/1/25

Journal of Cheminformatics

RESEARCH ARTICLE Open Access

From data to analysis: linking NWChem and Avogadro with the syntax and semantics of Chemical Markup Language

Wibe A de Jong[1*†], Andrew M Walker[2†] and Marcus D Hanwell[3†]

Abstract

Background: Multidisciplinary integrated research requires the ability to couple the diverse sets of data obtained from a range of complex experiments and computer simulations. Integrating data requires semantically rich information. In this paper an end-to-end use of semantically rich data in computational chemistry is demonstrated utilizing the Chemical Markup Language (CML) framework. Semantically rich data is generated by the NWChem computational chemistry software with the FoX library and utilized by the Avogadro molecular editor for analysis and visualization.

Results: The NWChem computational chemistry software has been modified and coupled to the FoX library to write CML compliant XML data files. The FoX library was expanded to represent the lexical input files and molecular orbitals used by the computational chemistry software. Draft dictionary entries and a format for molecular orbitals within CML CompChem were developed. The Avogadro application was extended to read in CML data, and display molecular geometry and electronic structure in the GUI allowing for an end-to-end solution where Avogadro can create input structures, generate input files, NWChem can run the calculation and Avogadro can then read in and analyse the CML output produced. The developments outlined in this paper will be made available in future releases of NWChem, FoX, and Avogadro.

Conclusions: The production of CML compliant XML files for computational chemistry software such as NWChem can be accomplished relatively easily using the FoX library. The CML data can be read in by a newly developed reader in Avogadro and analysed or visualized in various ways. A community-based effort is needed to further develop the CML CompChem convention and dictionary. This will enable the long-term goal of allowing a researcher to run simple "Google-style" searches of chemistry and physics and have the results of computational calculations returned in a comprehensible form alongside articles from the published literature.

Keywords: Chemical Markup Language, FoX, NWChem, Avogadro, Computational chemistry

Background

In chemistry, the key to successful multi-disciplinary integrated research is often the ability to couple the diverse sets of data obtained from a range of complex experiments and computer simulations to solve scientific problems that are intractable when only one technique is employed. In an ideal world any researcher in any discipline should be able to easily access, find and synthesise all the scientific data pertaining to the scientific question, molecular system or material being studied [1]. With this data the researcher has a knowledge base that has the potential to deliver new unexpected insights when all the information is brought together. Access to all scientific data relevant to the study at hand can also avoid repetition of previous experiments or simulations, and can serve as a starting point for generating new ideas for the design of alternative new approaches or molecular/materials systems, or can provide a framework for validation [2]. Increasing quantities of detailed data is gradually being made available to scientific users

* Correspondence: bert.dejong@pnnl.gov
†Equal contributors
[1]EMSL, Pacific Northwest National Laboratory, P.O. Box 999, Richland, WA 99352, USA
Full list of author information is available at the end of the article

in the scientific literature [3], and more widely in data repositories and other systems [4].

Raw data (for example the recorded NMR signal or calculated molecular orbitals) is only a subset of all the scientific data generated from an experiment or computer simulation. This acquired data gets processed and analyzed with a barrage of tools to extract the important observables, i.e. derived data, and help create the scientific interpretation. This derived data is most likely stored in the researcher's notebooks and sometimes forms part of scientific publications and presentations, but it rarely gets used to annotate the raw data. Observables are the common language at which computational chemistry and experimental communities interact, and these should be accessible as part of the scientific dialogue to make available data useful to the scientific community at large.

A key challenge is the need to remove technical barriers to access scientific data, and common data formats play a key role in this respect. In the experimental community raw data is often stored in a standardized format once it has been acquired (e.g. NeXus [5] or Scientific Data Exchange [6] using HDF5 [7]), lightly annotated with details of the experiment itself. While the experimental community has been working to develop and deploy data standards, this is less true in the computational chemistry community.

Historically, complete data sets from computer simulations, including input and all generated ASCII and binary output data files, have not been made widely available. These files are not generally stored in accessible data repositories or included as supplements with publications. Within computational chemistry and materials science there are multiple efforts to make a limited set of calculated observables available to the broader community. Examples include the Materials Project at MIT [8], CatApp at Stanford [9] the Computational Results Database at Washington State University [10] and the Computational Chemistry Comparison and Benchmark Database at the National Institute of Standards and Technology (NIST) [11]. The latter also links the information to the available experimental data available at NIST. Each of these efforts provides easy access to commonly used observables using diverse and non-standard data formats, but often with incomplete scientific data sets or lacking the important meta-data.

The key to effective integration, mining, reuse, and visualization of diverse data sources is to work within standardized and semantically rich data formats. The myriad of simulation data also stifles advances in the development of open-source data mining/search tools and visualization tools (such as Avogadro [12]) due to the significant efforts to develop and maintain translation infrastructures for all the data formats. A good example of standardization is the Crystallographic Information Framework, which includes the widely used Crystallographic Information File (CIF) [13] maintained by the International Union of Crystallography (IUCr). Naturally, many of the more recent efforts to develop standardised data formats have made use of the extensible markup language (XML) to define the format. The various XML standards provide a widely implemented framework for defining the basic syntax of data files, mechanisms for specifying the permitted structure and content of such files and common approaches to reading, writing, manipulating, normalising and validating standardised documents. These standards and tools enormously simplify the task of designing and deploying sharable data formats. Some relevant examples include the work of Gygi and co-workers, who developed a rudimentary XML-based standard for the interchange of simulation data [14] that is used in their framework for the validation and verification of electronic structure codes, called ESTEST [15]. Researchers at NIST led an effort to develop an XML standard for the interchange of materials information (MatML) [16].

Perhaps the most successful attempt to standardize the development of common language for chemistry is the Chemical Markup Language (CML), developed by Murray-Rust and Rzepa since 1995 [17-21]. CML provides the vocabulary needed to express a very wide range of chemical (and related physical) concepts in an XML document. This vocabulary is specified in an XML Schema definition (XSD) document that is deliberately permissive: it must be as the use of valid CML documents is extremely varied [22,23]. CML documents can act as interactive scholarly manuscripts [21], as supplementary data to support more traditional publication [3], as the primary data format for a range of experimental and computational studies [2,24], or as a means of data exchange within automated workflows [25-28]. For some of these applications it can be useful if the document structure and content is further restricted with additional constraints beyond those enforced by requiring validity as defined by the CML XSD specification. These additional constraints, termed CML Conventions, provide discipline-specific meaning to CML documents and reduce the development burden for document consumers and producers [29]. CML dictionaries provide additional fine-grained semantics with specific concepts identified by terms referenced from elements within CML documents [29]. Furthermore, these mechanisms impose additional good practice on document produces by requiring the inclusion of defined metadata to preserve provenance, and by insisting that all numerical data carries a machine readable specification of its units. It is the ability to impose strict syntax and defined semantics, and to validate these using well-understood

tools, that means XML in general and CML in particular is still the tool of choice for the storage and exchange of scientific data. While we imagine that alternative data representations, such as the JavaScript Object Notation (JSON), may be used in this context for efficient data exchange we do not foresee the development tools for powerful validation and semantic transformation that motivate the use of XML, as discussed below.

To enable different computational chemistry codes to interoperate it is key that all essential data is stored in a common format. Within the computational chemistry community various simulation codes, for example, MOLPRO, VASP, and Quantum-Espresso [30-33], have been adapted to produce a code specific XML output. Other projects have utilized components of CML as an enabling tool for data exchange, storage and processing. The Quixote project seeks to build a collection of tools to allow data from computational chemistry calculations to be stored, shared, organised and queried [25]. Key to this effort is the creation of a CML document for each calculation. This is then ingested into, and processed by, an instance of the Chempound database system. By contrast, the eMinerals [34] and Materials Grid [35] projects sought to develop automated scientific workflows for high-throughput computation in atomic-scale mineralogy and materials science. These workflows combined distributed grid computing [26,27] with tools to generate input files, extract and analyse output data, and create and store key items of metadata [28]. An important aspect of this work was the generation of a representation of the key data in a CML document. The approach taken to allow these documents to be easily produced on the myriad of systems that formed the distributed computing environment utilised by these projects was to directly generate CML as the computational chemistry application was executed. As the majority of such applications are written predominantly in Fortran, and the installed software on the computational resources was so varied, this involved the creation of a pure Fortran XML library called FoX [36,37], described below and in more detail by Murray-Rust and co-workers in another article in this special issue [38].

An alternative approach would be to utilize Python or a similar high level language to wrap the Fortran code and use features of the high level language to generate the CML. However the integration of Python and Fortran codes requires quite intrusive changes to the Fortran application including the development of interface routines for each piece of functionality in the code that needs to store output data in the CML document, a redesign of the existing build and test system, and potentially a change in the skills needed by members of the developer community. While we consider that this may be a viable approach if integration into a high-level programming environment was being undertaken for other reasons, we avoided this change merely for the purpose of generating an XML document. As new concepts, conventions, and dictionaries are defined within CML, new common interfaces can be developed in the FoX library and used by the computational chemistry codes. FoX and its interfaces were used to allow solid-state simulation codes such as SIESTA [39] and GULP [40] to directly output CML without introducing additional dependencies into the applications' compilation process. Very recently, developers of the TURBOMOLE package reported the use of CML and the interfaces in the FoX library to store a subset of their output data needed to develop a database for computational data [41].

In this paper the further development of a FoX based infrastructure to produce semantically rich CML documents with the NWChem computational chemistry software [42] will be described. NWChem is one of US Department of Energy's open-source computational chemistry software packages, developed at the Environmental Molecular Sciences Laboratory (EMSL) National User Facility located at Pacific Northwest National Laboratory. In addition, some developments of the CML language for computational chemistry will be discussed in some detail. Furthermore, the reading, ingestion and visualization of the CML documents generated by NWChem will be demonstrated in Avogadro, an open-source, cross-platform molecular editor and analysis tool written mainly in C++ [13].

Methods

Generating semantically rich data with NWChem

The NWChem software like many other computational chemistry software applications produces various data files during a simulation. These data include a human readable descriptive output file and binary files potentially containing from tens of megabytes to multiple gigabytes of additional data (such as molecular orbitals and molecular dynamics trajectories). One approach to the creation of a CML document to describe an NWChem calculation is to post-process the existing output data files and convert these into CML. This is the path taken by the Quixote project where Java Universal Molecular Browser for Objects (JUMBO) converters are used to transform the human readable output files to CML documents that can be digested by tools such as Chempound [25]. The major disadvantage of the use of converters is that they need to be continuously modified and maintained because the output files they read have a tendency to change, for example when new functionality is added to NWChem. The NWChem developer modifying the codebase in this way has no way of knowing if the change causes external tools to fail and, probably,

has no reason to care. In addition, the content extracted from the simulation is limited to what is contained in these text based output files. Additional information held within associated large binary files, for example molecular orbitals or time evolution data, is lost unless this data also gets post-processed. Both of these issues are addressed by our approach as described below.

In addition to the text and binary files, NWChem has a built-in infrastructure to produce an additional ASCII file that contains key-value pairs. This file serves as input for the EMSL developed Extensible Computational Chemistry Environment (ECCE) graphical user interface and visualizer [43]. Ironically, ECCE reads this file and translates it into an application specific XML format that is used internally. This means that both the JUMBO converters and ECCE use multiple steps to obtain a similar final XML product.

As part of the current work, a prototype CML writer capability in NWChem has been developed that writes a CML file during the simulation run. To minimise intrusive modifications to the code base, the implementation builds upon the infrastructure in NWChem for writing key-value pairs into the ECCE ASCII file. This infrastructure inherently already opens a data file at start up, writes data to the file, and closes the file when the simulation ends. Instead of writing a standard ASCII text file, NWChem's ECCE infrastructure has been modified to make use the FoX library to create the CML document. This makes more data available to the CML writing machinery than is written to the output file designed for human consumption. We have benchmarked the performance impact of writing CML files on a larger (6 minute runtime) version of the example case used in this paper and find the run time to be increased by less than 0.5% (the impact falls below 0.1% for even larger calculations). The CML writing capability is being integrated into the main NWChem development tree and will become a standard output format in a future release.

As discussed elsewhere in this issue [38], the FoX library permits the reading, writing and manipulation of arbitrary XML documents in a Fortran-only environment. As expected, FoX will not permit the creation of an XML document that is not well-formed. As well as the general-purpose interfaces, FoX_wcml provides a specialised mechanism for generating CML output tailored to computational chemistry applications. This interface, which was used for the majority of the CML generation from NWChem, does not attempt to allow the creation of any arbitrary CML document but instead focuses only on the needs of the majority of expected use cases in computational chemistry and to a large extent imposes adherence to the relevant conventions. For example, users of FoX_wcml are not able to add atoms to a molecule without specifying both position and element name, avoid declaring (and using) the relevant namespace for CML, or introduce numerical quantities without specifying units. Data passed into the FoX_wcml interface is accepted in a format expected to be present inside of computational chemistry applications but some data conversion is still sometimes needed. Figure 1 shows an example of a fragment of the CML document that can be generated from NWChem with this implementation. In Additional file 1 we provide a full example of a CML document produced by our current implementation. We note that this passes the tests implemented by the CML validation service [44].

A side effect of the constrained interface is that new ideas for the representation of output data cannot easily be generated with FoX_wcml. However, the generic FoX_wxml interface can be used in concert with FoX_wcml to produce additional output in an already open CML document. We made use of this feature to expand the FoX_wcml interface to include an ability to write molecular orbitals to the CML document. Molecular orbitals are currently not well defined in the CML language. Appendix 1 outlines a draft CML computational chemistry dictionary for molecular orbitals, and shows a formatted example that can be used as a template to develop a molecular orbital convention in the future. We also enabled the ability to include a representation of the main NWChem ASCII file input parameters within the output document. Such 'echoing' back of the exact code input in the output stream is very common in computational chemistry and is essential to preserve data provenance (for example, if the input files are inadvertently lost or modified). An example of the data inside a CML document produced by NWChem is provided in Figure 1 and a more formal definition of the proposed CML format echoing computational input is included in Appendix 2 (where we assume only ASCII data will be stored: arbitrary binary files with, for example, wave functions are out of scope). This new functionality in the FoX library is publically available in the development source control system and will be included in the next formal release of the software.

Visualisation and analysis of semantically rich NWChem data with Avogadro

In order to read in the CML produced by NWChem or any other computational chemistry code, a new reader was developed for Avogadro in the OpenQube library. As discussed briefly in another publication in this journal issue [12], OpenQube was developed to address the requirements of normalization of data produced by codes employing Gaussian type orbitals for calculations. It was already able to recognize the data available in output from several other quantum codes, with the new

```
<?xml version="1.0" encoding="UTF-8"?>
<cml convention="convention:compchem" fileId="prop_h2o.cml" xmlns="http://www.xml-cml.org/schema" xmlns:compchem="http://www.xml-cml.org/dictionary/compchem/"
xmlns:nwchem="http://www.nwchem-sw.org/dictionary/nwchem/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:fpx="http://www.uszla.me.uk/fpx"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:convention="http://www.xml-cml.org/convention" xmlns:unit="http://www.xml-cml.org/unit/si"
xmlns:fmisc="http://www1.gly.bris.ac.uk/~walker/namespaces/foxmisc">
  <metadata name="fmisc:UUID" content="3dcb5fb0-3933-11e2-7262-f64ed3ff2cb8"/>
  <module title="NWChem simulation" dictRef="compchem:jobList">
    <module dictRef="nwchem:Input" role="LexicalFile">
      <metadataList>
        <metadata name="fmisc:filename" content="prop_h2o.nw"/>
      </metadataList>
      <scalar dataType="xsd:string">echo</scalar>
      <scalar dataType="xsd:string">start prop_h2o_run</scalar>
      <scalar dataType="xsd:string">title h2o</scalar>
      <scalar dataType="xsd:string">ecce_print prop_h2o.cml</scalar>
      <scalar dataType="xsd:string"></scalar>
      <scalar dataType="xsd:string">geometry units au nocenter</scalar>
      <scalar dataType="xsd:string">o       .00000000      .00000000      .11786656</scalar>
      <scalar dataType="xsd:string">h       .00000000     1.84118838     -.93531364</scalar>
      <scalar dataType="xsd:string">h       .00000000    -1.84118838     -.93531364</scalar>
      <scalar dataType="xsd:string">end</scalar>
      <scalar dataType="xsd:string"></scalar>
      <scalar dataType="xsd:string">basis</scalar>
      <scalar dataType="xsd:string">  * library cc-pvdz</scalar>
      <scalar dataType="xsd:string">end</scalar>
      <scalar dataType="xsd:string">charge 0</scalar>
      <scalar dataType="xsd:string"></scalar>
      <scalar dataType="xsd:string">task scf energy</scalar>
      <scalar dataType="xsd:string"></scalar>
    </module>
    <module title="NWChem runtime" dictRef="compchem:environment">
      <propertyList>
        <property dictRef="compchem:program">
          <scalar dataType="xsd:string">NWChem</scalar>
        </property>
        <property dictRef="compchem:programVersion">
          <scalar dataType="xsd:string">6.1</scalar>
        </property>
        <property dictRef="compchem:runDate">
          <scalar dataType="xsd:string">Wed Nov 28 16:11:18 2012</scalar>
        </property>
        <property dictRef="compchem:numProc">
          <scalar dataType="xsd:integer" units="unit:none">1</scalar>
        </property>
      </propertyList>
    </module>
    <module title="task_energy" dictRef="nwchem:task_energy">
      <molecule id="geometry">
        <atomArray>
          <atom elementType="O" x3="0.000000000000e0" y3="0.000000000000e0" z3="6.237229959749e-2" id="o"/>
          <atom elementType="H" x3="-9.743150018186e-1" y3="0.000000000000e0" z3="-4.949467013395e-1" id="h"/>
          <atom elementType="H" x3="9.743150018186e-1" y3="0.000000000000e0" z3="-4.949467013395e-1" id="h"/>
        </atomArray>
      </molecule>
      <module title="scf" dictRef="nwchem:scf">
        <list dictRef="compchem:basisSet" id="ao basis">
          <list dictRef="compchem:basisSetContractions" id="basisSetContractions1">
            <scalar dataType="xsd:string" id="atomType" dictRef="compchem:atomType">o</scalar>
            <scalar dataType="xsd:string" id="elementType" dictRef="compchem:elementType">O</scalar>
            <scalar dataType="xsd:string" id="basisSetType" dictRef="compchem:basisSetType">orbital</scalar>
            <scalar dataType="xsd:string" id="basisSetTitle" dictRef="compchem:basisSetTitle">cc-pvdz</scalar>
            <scalar dataType="xsd:string" id="basisSetHarmonicType" dictRef="compchem:basisSetHarmonicType">cartesian</scalar>
```

Figure 1 First section of CML validated XML data, generated with NWChem using the FoX library.

computational chemistry CML reader being one of the fastest to develop using the compact PugiXML library as a lightweight XML parsing engine. The example NWChem CML data in this paper was read into Avogadro using the new reader, and from that point on behaved as other data files.

An experimental branch is available for Avogadro that can read in important electronic structure information from NWChem, display it to the user and even produce further input for NWChem based on geometries from previous calculation results. This allows Avogadro to leverage the semantic information stored in the CML document with minimal code changes. Figure 2 shows an integrated visualization of the molecular orbital isosurface and the nuclear magnetic resonance (NMR) shielding tensors calculated by Avogadro using the data read in by this new reader, successfully demonstrating an end-to-end semantically enriched workflow. The visualization of both properties in more complex molecules can provide new insights into molecular bonding.

The new reader was written using the PugiXML C++ XML parser library, and took the approach of using an efficient Document Object Model (DOM) to represent the CML document, and specific functions to read in and process XML nodes of interest. One of the major advantages of this approach over other readers was the minimal amount of parsing code as it was possible to rely upon the standard XML parsing routines to find the nodes that contained data that needed to be read in. More work will be required to generalize the approach taken as a single CML document can contain a rich computational experiment with multiple steps. Additional interfaces must be developed in Avogadro to go beyond a simple view of an output file containing one result to that of a rich document that can contain multiple steps with links.

A major advantage of the CML reader is that of extensibility; where the CML document can contain elements that this version of the reader does not necessarily understand. Most other readers are somewhat fragile,

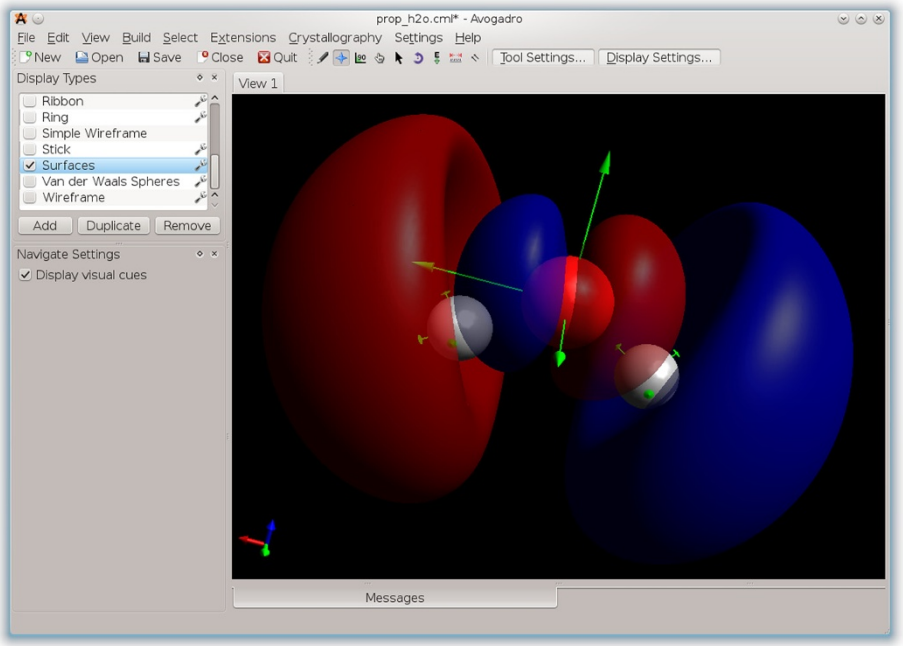

Figure 2 Visualization of molecular orbital isosurface calculated by Avogadro using the NWChem CML data, with NMR tensor projections and magnitudes represented as arrows.

and require updates when the computational code adds extra elements to its output. This reader is able to scan the CML document for the tags and attributes it understands, if more tags/attributes are added in the future the reader will still be capable of understanding the document, with updates to the reader adding support for the new elements. This allows for the development of a much more robust approach to reading and transforming the data in the Avogadro application that was simply not possible in the other readers developed.

Results and discussion

Our experience of the use of FoX for introducing CML output into NWChem has reinforced our understanding of various best-practice guidelines for the integration of FoX to large-scale computational chemistry simulation codes. By modifying NWChem to directly generate CML we avoid some of the potential difficulty with external converters discussed above. Any NWChem developer modifying the codebase will see, directly in the code being modified, the subroutine calls that generate the CML document. This, in itself, alerts the developer of the potential for their changes to break the generation of CML (or, at least, for the potential to break some unknown feature of the code). This cue is absent in the case of external converters that parse an output file only designed for human consumption. If this hint to developers is insufficient the use of FoX provides a second line of defence against breakage of the CML output: such damage will result in the NWChem code failing to compile or run simple test cases. The developer will thus be motivated to correct the output immediately, hopefully before checking the change into their version control system. Again, this opportunity is absent in the case of the use of external file converters.

For comparatively simple applications the FoX_wcml interface should be sufficiently clear to allow calls to be made directly from the main body of the simulation code. However, as the simulation code becomes larger it becomes important to isolate the calls to the FoX library within a 'glue' layer (often a single Fortran 90 module) within the application. This technique was adopted in NWChem. As is typical, the layer brings together groups of calls to FoX routines within a higher-level interface adapted for the data structures used by the application and isolates the necessary (and often simple but tedious) data conversion tasks from the important numerical computation. This isolation helps minimise the possibility that an unrelated change to the main numerical parts of the code would cause the CML output to fail (for example, by attempting to generate an ill-formed document resulting in FoX terminating execution). This can be particularly important if the code is developed by a large team as it is possible that only a few developers understand the constraints of the FoX API and XML more generally. Placing the FoX calls in an isolated

module also makes it comparatively easy to make the generation of CML output a runtime or compile time option allowing, for example, some of the more egregious compiler bugs to be avoided when deploying on systems with a limited choice of Fortran compiler.

Once a computational chemistry code like NWChem produces CML data, tools are required to read in and analyse this data. One such tool is Avogadro, which was extended to read in the CML data files produced by NWChem and to display the molecular orbitals using the same library functionality written for other output formats. The major advantage of this approach over those previously taken is in the clear definition of what each term means, and how it should be used. The file reader can also be much simpler as it makes use of standard C++ libraries to read in and scan the XML document for the expected tags, allowing for a reader that is likely to continue working as expected even as new output types are added to the XML document.

There are of course other options for data storage, with formats such as JSON being used in an increasing number of projects for data exchange. The form and syntax of JSON is much simpler than XML, allowing for smaller and simpler parsers but there are also several drawbacks, which make XML a better choice for data exchange between loosely coupled components. JSON schemas are in their infancy, with no support for namespaces and nascent validation tools, all of which can be very helpful when composing complex documents incorporating elements from several sources. This leads to software that must generally be more tightly coupled to the data representation chosen in a particular implementation. This allows for XML based documents, such as CML, to compose multiple concepts such as scientific units defined by the wider community in a dedicated namespace composed with chemistry specific terms in the CML namespace, such as atomic positions and the ground state energy of the system. Due to the similarities in base representation, round-tripping between CML and JSON representations is not very difficult, and simple mapping schemes can be developed for applications where JSON is preferred as the underlying storage mechanism or lightweight data exchange container in more tightly coupled systems. XML documents retain a much richer semantic structure, which is also more readily converted to Resource Description Framework (RDF) for further and more generalized consumption in semantic frameworks.

Not all important data produced by a computational chemistry simulation is suitable to be stored and fully expressed in an XML format. Examples include the generally large data files storing molecular orbitals, densities on a grid, or time evolution files generated by molecular dynamics simulations. Work is underway to develop a two-layer data infrastructure through the integration of the eXtensible Data Model and Format (XDMF) [45] with the CML. Using XDMF allows semantically rich data, such as the common observables discussed above, to be stored in a searchable framework, while all the large data (such as orbitals, trajectories, etc.) will be stored in the compact HDF5 format [7].

Chemical Markup Language for computational chemistry

Coupling diverse sets of data requires the development of a common language to describe observables that should span experimental and computational technologies. CML [18] provides a high level language for chemistry while the dictionaries and, to a lesser extent, the conventions, offer a starting point for the development of rich semantically enabled tools. Various fields, such as computational chemistry and NMR, XPS, and other experimental capabilities, are developing conventions and dictionaries using CML. We believe that these dictionaries can begin to be linked to form a more unified ontology [46,47] to start to formally define relationships between terms in the different fields of chemistry. Such an effort will provide the common semantically rich language needed to integrate diverse sets of data. However, the use of CML dictionaries is currently underdeveloped. Most of the current technology only requires that dictionary references are used and that they uniquely identify a particular concept. Increasingly these concepts are defined by entries in actual dictionaries and this is an important aspect of providing shared meaning but defining computationally accessible links between concepts is also crucial. This requires both the links to be present in the dictionaries and tools to exploit these links, probably via a transformation to RDF, to be developed.

Looking forward, one of the major challenges for the community will be to define a unified convention and dictionary for the electronic wave function. The current CML computational chemistry convention and dictionary for Gaussian basis sets and effective core potentials is based on the XML format defined by the EMSL Basis Set Exchange. [48] Some work on defining a standard format for plane wave basis sets has been done by Gygi et al. [15]. The EMSL Basis Set Exchange has the potential, especially when expanded to describe plane wave basis sets, to serve as a reference database on its own.

Conclusions

We have developed an end-to-end use of semantically rich data in computational chemistry within the Chemical Markup Language (CML) framework. NWChem was relatively easily modified to use the FoX library to produce well-formed and valid CML documents that conform to recent conventions and are semantically rich. Draft dictionary entries as well as a proposed CML CompChem format for describing molecular orbitals were developed and included in the FoX library. The

FoX library was expanded to provide semantics that enable the scientific application to represent the raw ASCII input file(s). A new CML reader developed for Avogadro was used to read and visualize the NWChem computational chemistry CML data. We briefly discussed the need for a community-based development of a unified computational chemistry convention and dictionary, and outlined some of the challenges to its development.

The long-term goal is the development of a common language and data infrastructure for the chemistry community that will enable machines to easily process data generated by computational chemistry tools. By clearly embedding the semantics of the document with the raw data we hope to limit or eliminate ambiguity when data documents are indexed, searched or otherwise processed and enable scientific discovery through simple and robust interfaces. We envisage a situation where a researcher, prior to embarking on a calculation, enters key parameters in a search tool, perhaps resembling the Google interface, and that this would return rendered and understandable representations of the output of previous calculations along with links to the published literature. By embedding the semantics inside the data document from their creation we remove the need for excessive inference of the semantics by the indexing system. A useful side effect of this effort, demonstrated in this contribution, is the ease that the technology can be used to link existing simulation and visualisation tools in the computational chemistry domain. At present efforts are underway to integrate both experimental and NWChem's computational data at the EMSL, a national user facility of the U.S. Department of Energy's Office of Biological and Environmental Research. Eventually, all CML data from the facility's computing capabilities will be available in a searchable data archive and researchers will be able to gain new scientific insight through access and visualization of complex sets of simulation and experimental data.

Appendix 1: Molecular orbitals dictionary entries

As discussed in the main text, in the process of adding CML output to NWChem we developed a draft of dictionary entries with descriptions and a proposed CML CompChem format for describing molecular orbitals. These dictionary entries and descriptions will be integrated in the CML CompChem dictionary to a reference to the semantic meaning of the concepts in CML documents that include the specification of molecular orbitals. This dictionary is already used in a range of applications that require this information internally (e.g., for unit validation) or to provide human readable descriptions of the terms (e.g. to provide help text in documents transformed into HTML). By being included in the dictionary, the new terms will also become available on the web.

Dictionary entries

Molecular Orbitals (dictRef: molecularOrbitals)

Definition: CML list container for all information related to one set of molecular orbitals.
Description: A set of wavefunctions describing all electrons in a system of atoms.
Data Type: molecularOrbitals is of data type cml:list
Unit Type: molecularOrbitals has unit type unitType: none

Atomic Basis Descriptions (dictRef: atomicBasis Descriptions)

Definition: A cml array containing the descriptions of atomic basis functions.
Description: Atomic basis functions constructed from linear combinations of Gaussian functions that describe atomic orbitals and form the basis for the molecular orbital in the format < atom number: atom name, shell type>, and of the type xsd:string. Shell type refers to the angular momentum of the basis function (s, px, py, pz, dxx, dxy, dxz, dyy, dyz, dzz, etc. for cartesian basis functions and s, px, py, pz, d −2, d −1, d 0, d 1, d 2, etc. for spherical basis functions).
Data Type: atomicBasisDescriptions is of data type cml:array
Unit Type: atomicBasisDescriptions has unit type unitType:none

Molecular Orbital (dictRef: molecularOrbital)

Definition: A cml list container for one molecular orbital.
Description: Mathematical representation of the wavefunction of an electron in a system of atoms described by a linear combination of atomic basis functions or atomic orbitals.
Data Type: molecularOrbital is of data type cml:list
Unit Type: molecularOrbital has unit type unitType:none

Orbital Energy (dictRef: orbitalEnergy)

Definition: Total energy of the molecular orbital.
Description: Energy of the electron described by the molecular orbital in Hartrees.
Data Type: orbitalEnergy is of data type xsd:double
Unit Type: orbitalEnergy has unit type unitType:energy

Orbital Symmetry (dictRef: orbitalSymmetry)

Definition: Point group symmetry of the molecular orbital.
Description: Symmetry character of the molecular orbital within the point group symmetry of the molecule.

Data Type: orbitalSymmetry is of data type xsd:string
Unit Type: orbitalSymmetry has unit type unitType:none

Orbital Spin (dictRef: orbitalSpin)

Definition: Spin of the orbital.
Description: Spin symmetry of the molecular orbital as either “alpha” or “beta”.
Data Type: orbitalSpin is of data type xsd:string
Unit Type: orbitalSpin has unit type unitType:none

Orbital Occupancy (dictRef: orbitalOccupancy)

Definition: Occupancy of molecular orbital.
Description: Number of electrons occupying the molecular orbital. The value will range from 0.0 to 2.0 electrons. When the orbital spin is defined as either alpha or beta, the maximum occupation can be 1.0.
Data Type: orbitalOccupancy is of data type xsd:double
Unit Type: orbitalOccupancy has unit type unitType: none

Atomic Basis Function Composition of Molecular Orbital (dictRef: aoVector)

Definition: The cml array containing atomic basis function coefficients.
Description: A cml array of xsd:double containing the coefficients of the linear combination of atomic basis functions that describe the molecular orbital.
Data Type: aoVector is of data type cml:array
Unit Type: aoVector has unit type unitType:none

Example of molecular orbitals

Below an example of the molecular orbitals CML format. Shown is the header and first two orbitals of the H2 molecule:

```
<list dictRef="molecularOrbitals" id="nmrrun.movecs">
  <array size="10" delimiter="|" dataType="xsd:string" id="aoDescriptions"
dictRef="atomicOrbitalDescriptions">1 H  s      |1 H  s      |1 H  px     |1 H  py     |1 H
pz     |2 H  s      |2 H  s      |2 H  px     |2 H  py     |2 H  pz
  </array>
  <list dictRef="molecularOrbital" id="molecularOrbital1">
    <scalar dataType="xsd:double" dictRef="orbitalEnergy">-4.371860531460e-1</scalar>
    <scalar dataType="xsd:string" dictRef="orbitalSymmetry">a</scalar>
    <scalar dataType="xsd:double" dictRef="orbitalOccupancy">2.000000000000e0</scalar>
    <array size="10" dataType="xsd:double" dictRef="aoVector">3.306455447679e-1
3.187524188824e-1 2.591476635736e-18 9.610825759597e-18 -2.104953907844e-2
3.306455447679e-1 3.187524188824e-1 3.164500292222e-18 7.783890197431e-18
2.104953907844e-2
    </array>
  </list>
  <list dictRef="molecularOrbital" id="molecularOrbital2">
    <scalar dataType="xsd:double" dictRef="orbitalEnergy">3.754134049814e-2</scalar>
    <scalar dataType="xsd:string" dictRef="orbitalSymmetry">a</scalar>
    <scalar dataType="xsd:double" dictRef="orbitalOccupancy">0.000000000000e0</scalar>
    <array size="10" dataType="xsd:double" dictRef="aoVector">2.828457867209e-1
8.018372300029e-1 1.646912668252e-16 -4.322183081884e-17 1.409042171552e-2 -
2.828457867209e-1 -8.018372300029e-1 -1.615375887293e-16 4.120378217274e-17
1.409042171552e-2
    </array>
  </list>
</list>
```

Appendix 2: Representing input data

In the process of adding CML output to NWChem we found that it is important to include a direct representation of the NWChem input parameters within the CML output for provenance. Although the CML Schema has sufficient flexibility to allow this none of the existing conventions contain a suitably defined mechanism for this task. In this appendix we outline a suitable microformat for the representation of Fortran input files in computational chemistry. The intention is that this representation forms part of the CML CompChem convention but is sufficiently flexible to be reused in other contexts. We note that a textural representation of input files does not offer the same degree of semantic interoperability as including input data in a fully marked-up format (i.e. as CML parameters with dictionary references and explicit units). However, the task of recreating input files from such data is a non-trivial problem that has, thus far, received little attention. The microformat defined here can be viewed as an essential intermediate step to allow a complete representation of the input data to be stored in the meantime. Applications making use of the microformat should additionally report input data semantically, as CML parameters.

The microformat is designed to allow easy storage and retrieval of the content of one or more files, or of data read from another file-like source (e.g. from standard input). The aim is to make it possible to reconstruct the input using a simple streaming parser without the need to build an in-memory representation of the CML document (in order to handle cases of very large files) even if the XML document has been modified (e.g. if the document has been subject to Canonicalization [49]). The approach is also designed to make data recovery using an XSL transform straightforward. File metadata such as the original filename is also accessible. We assume that only ASCII data must be stored, arbitrary binary files are out of scope as are XML documents and non-ASCII textural data.

Format for the representation of input data

This specification uses the namespaces and prefixes to indicate those namespaces as outlined in Table 1.

Table 1 Namespaces and namespace prefixes used in the representation of input file data

Prefix	Namespace URI	Description
cml	http://www.xml-cml.org/schema	Chemical Markup Language elements
convention	http://www.xml-cml.org/convention/	Standard Chemical Markup Language convention namespace
compchem	http://www.xml-cml.org/dictionary/compchem/	CompChem Dictionary namespace

Element names, and where necessary, attribute values, are given as QNames (i.e. in the form prefix:localName) and the prefix must be in scope and bound to the appropriate namespace URI. The data format is defined as follows:

(a) The cml:module element may be used as a container for one or more input files. If used this outer module must have a dictRef attribute with the value 'compchem:inputFileList'. There may be a title attribute giving this information in brief human readable form. Rationale: It is important to be able to easily locate input file data in large documents. By enclosing all such data in a single parent element with a uniquely defined dictionary reference a low-memory streaming parser can extract the data without the need for complex document manipulation.

(b) Data for each input file must be contained in its own cml:module element. Each such element must have a dictRef attribute with the value "compchem:inputFile" and may have a title attribute. It should be possible to convert the contents of each such module to a single ASCII file used as input to an atomistic simulation application without reference the rest of the XML document. Rationale: Some applications read input data from multiple input files. It must be possible to reconstruct each of these files.

(c) Each module element described in (b) should contain a single cml:metadataList element with zero or more number of cml:metadata child elements. These elements are intended to contain metadata that can be used help the (re)creation of the calculation's input data. It is intended that a wider range of information could be conveyed in this way but we pay particular attention here to the file name of the input data file represented by the parent module. The file name should be stored as the content attribute of a metadata element with the name attribute being compchem:inputFileName. Rationale: Associated with each file are various items of metadata. While not all file-like objects will have a name where it is present the file name can be used to recreate the input data needed by the application (which, sometimes, must be contained in files with a fixed name).

(d) Each module element described in (b) should contain one or more cml:scalar child elements. These should have the attribute 'dataType' with the value 'xsd:string'. Text content of these elements should each correspond to a single (XML encoded) line of the input file. All white space should be preserved but line-ending characters should be removed. The order of the lines in the input file must correspond to the document order of the cml:scalar elements as defined in section 5 of the XPath specification [50]. Empty lines should be represented by empty cml:scalar elements. Whitespace, including tabulation characters, multiple spaces and spaces at the start and end of lines should be preserved unaltered unless it is known that the application generating the CML document is insensitive to such changes. Rationale: This approach allows the location of each line of input via an XPath expression without the risk of modification due to normalisation of the document. Whitespace characters must be retained unaltered as, for some Fortran applications, the exact amount, form and location of such characters can be significant.

The CML file shown in Figure 1 and included in the Additional file 1 illustrates the use of this microformat.

Reading and writing file data

A small number of new subroutines have been added to the FoX_wcml module to permit Fortran applications to easily create representations of their input files in an output CML document using the format outlined above. Two methods are provided with the most appropriate being dependent on the design of the application. In the first approach a single subroutine, cmlDumpInputDeck, is called with an array of file names as input arguments. In turn each file is opened, its contents are written to the CML document in the appropriate form, before the file is closed. The convoluted nature of file handling in Fortran combined with the way that some applications read their input data means that this approach is not always available (for example, if the input file is held open for the duration of the calculation, or if data is read from standard input) so an alternative interface with five subroutines (cmlStartInputDeckList, cmlStartInputDeckFile, cmlAddInputDeckLine, cmlEndInputDeckFile and cmlEndInputDeckList) is provided. The FoX documentation provides full details of how to use these two interfaces [51].

Finally, we provide a trivial example script to show one way of recreating input files given a CML document. This uses XPath [50] and python and is provided in Additional file 1.

Additional file

Additional file 1: The following additional data are available with the online version of this paper. Additional data file prop_h2o.cml is a full example of a validated CML file produced by the modified NWChem version discussed in this paper, running the input file entitled prop_h2o.nw. The full output file prop_h2o.output is also provided. Additional data file input_files_example.xml is a simple example of the proposed input data format described in Appendix 2 and produced with new features of the FoX_wcml library. Additional data file extract_input_files.py is a straightforward python script that can be used to reproduce input data files stored in a CML document following the format described in Appendix 2.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
WAJ and AMW carried out the work on the implementation of FoX in NWChem, while MDH carried out the work on Avogadro. All authors drafted, read and approved the final manuscript.

Acknowledgements
A portion of the research was supported by EMSL, a national scientific user facility sponsored by the U.S. Department of Energy's (DOE) Office of Biological and Environmental Research and located at Pacific Northwest National Laboratory (PNNL). PNNL is operated for the DOE by the Battelle Memorial Institute under contract DE-AC06-76RLO-1830. This work has also received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement number 240473 "CoMITAC". MDH would like to thank the US Army Engineer Research and Development Center for funding under contract W912HZ-12-C-0005.

Author details
[1]EMSL, Pacific Northwest National Laboratory, P.O. Box 999, Richland, WA 99352, USA. [2]School of Earth Sciences, University of Bristol, Wills Memorial Building, Queen's Road, Bristol BS8 1RJ, UK. [3]Department of Scientific Computing, Kitware, Inc., 28 Corporate Drive, Clifton Park, NY 12065, USA.

Received: 4 January 2013 Accepted: 17 April 2013
Published: 24 May 2013

References

1. Murray-Rust P: **Chemistry for everyone.** *Nature* 2008, **451:**648–651.
2. Downing J, Murray-Rust P, Tonge AP, Morgan P, Rzepa HS, Cotterill C, Day N, Harvey MJ: **SPECTRa: The deposition and validation of primary chemistry research data in digital repositories.** *J Chem Inf Model* 2008, **48:**1571–1581.
3. Rzepa HS: **The past, present and future of scientific discourse.** *J Cheminfo* 2011, **3:**46.
4. Marcial LH, Hemminger BM: **Scientific data repositories on the Web: An initial survey.** *J Am Soc Inf Sci* 2010, **61:**2029–2048.
5. Maddison DR, Swofford DL, Maddison WP: **Nexus: an extensible file format for systematic information.** *Syst Biol* 1997, **46:**590–621.
6. *Scientific Data Exchange.* http://www.aps.anl.gov/DataExchange/.
7. HDF Group: *Hierarchical data format version 5, 2000–2010.* http://www.hdfgroup.org/HDF5.
8. Jain A, Hautier G, Moore C, Ong SP, Fischer C, Mueller T, Persson KA, Ceder G: **A high-throughput infrastructure for density functional theory calculations.** *Comp Mat Sci* 2011, **50:**2295–2310.
9. Hummelshoj F, Abild-Pedersen F, Studt F, Bligaard T, Norskov J: **CatApp: A Web application for surface chemistry and heterogeneous catalysis.** *Angew Chem Int Ed* 2012, **51:**272–274.
10. Feller D: **The role of databases in support of computational chemistry calculations.** *J Comp Chem* 1996, **17:**1571–1586.
11. Johnson RD III: *NIST Computational Chemistry Comparison and Benchmark Database, NIST Standard Reference Database Number 101, Release 15b.* 2011. http://cccbdb.nist.gov.
12. Hanwell MD, Curtis DE, Lonie DC, Vandermeersch T, Zurek E, Hutchison GR: **Avogadro: An advanced semantic chemical editor, visualization, and analysis platform.** *J Cheminfo* 2012, **4:**17.
13. McMahon B: **Applied and implied semantics in crystallographic publishing.** *J Cheminfo* 2012, **4:**19.
14. *XML standards for simulation data.* http://www.quantum-simulation.org.
15. Yuan G, Gygi F: **ESTEST: a framework for the validation and verification of electronic structure codes.** *Comput Sci Disc* 2011, **3:**015004.
16. *MatML Standard.* http://www.matml.org.
17. Murray-Rust P, Townsend JA, Adams SE, Phadungsukanan W, Thomas J: **The semantics of Chemical Markup Language (CML): dictionaries and conventions.** *J Cheminfo* 2011, **3:**43.
18. Murray-Rust P, Rzepa HS, Wright M: **Development of Chemical Markup Language (CML) as a system for handling complex chemical content.** *New J Chem* 2001, **25:**618–634.
19. Murray-Rust P, Rzepa HS: **Chemical markup, XML, and the Worldwide Web. 1. Basic principles.** *J Chem Inf Comp Sci* 1999, **39:**928–942.
20. Murray-Rust P, Rzepa HS, Wright M, Zara S: **A universal approach to web-based chemistry using XML and CML.** *Chem Comm* 2000:1471–1472. doi:10.1039/B002483J.
21. Murray-Rust P, Rzepa HS: **CML: Evolution and design.** *J Cheminf* 2011, **3:**44.
22. Townsend J, Murray-Rust P: **CMLLite: a design philosophy for CML.** *J Cheminf* 2011, **3:**39.
23. Murray-Rust P, Rzepa HS: **Chemical markup, XML, and the World Wide Web. 4. CML schema.** *J Chem Inf Comp Sci* 2003, **43:**757–772.
24. Wakelin J, Murray-Rust P, Tyrrell S, Zhang Y, Rzepa HS, García A: **CML tools and information flow in atomic scale simulations.** *Mol Sim* 2007, **31:**315–322.
25. Adams S, de Castro P, Echenique P, Estrada J, Hanwell MD, Murray-Rust P, Sherwood P, Thomas J, Townsend J: **The Quixote project: Collaborative and Open Quantum Chemistry data management in the Internet age.** *J Cheminfo* 2011, **3:**38.
26. Bruin RP, White TOH, Walker AM, Austen KF, Dove MT, Tyer RP, Couch PA, Todorov IT, Blanchard MO: **Job submission to grid computing environments.** *Concurrency Computat: Pract Exper* 2008, **20:**1329–1340.
27. Walker AM, Bruin RP, Dove MT, White TOH, Kleese-van Dam K, Tyer RP: **Integrating computing, data and collaboration grids: the RMCS tool.** *Phil Trans R Soc A* 2009, **367:**1047–1050.
28. Kleese-van Dam K, James M, Walker AM: **Integrating data management and collaborative sharing with computational science processes**. In *Handbook of Research on Computational Science and Engineering: Theory and Practice Volume 1.* Edited by Leng J, Sharrok W. Hershey, Pennsylvania: IGI Global; 2011:506–538.
29. Murray-Rust P, Townsend J, Adams SE, Phadungsukanan W, Thomas J: **The semantics of Chemical Markup Language (CML): dictionaries and conventions.** *J Cheminf* 2011, **3:**43.
30. Giannozzi P, Baroni S, Bonini N, Calandra M, Car R, Cavazzoni C, Ceresoli D, Chiarotti GL, Cococcioni M, Dabo I, Dal Corso A, Fabris S, Fratesi G, de Gironcoli S, Gebauer R, Gerstmann U, Gougoussis C, Kokalj A, Lazzeri M, Martin-Samos L, Marzari N, Mauri F, Mazzarello R, Paolini S, Pasquarello A, Paulatto L, Sbraccia C, Scandolo S, Sclauzero G, Seitsonen AP, Smogunov A, Umari P, Wentzcovitch RM: **Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials.** *J Phys Condens Matter* 2009, **21:**395502.
31. Gordon MS, Schmidt MW: **Advances in electronic structure theory: GAMESS a decade later**. In *Theory and Applications of Computational Chemistry, the first forty years.* Edited by Dykstra CE, Frenking G, Kim KS, Scuseria GE. Amsterdam: Elsevier; 2005:1167–1189.
32. Werner H-J, Knowles PJ, Knizia G, Manby FR, Schütz M: **Molpro: a general-purpose quantum chemistry program package.** *WIREs Comp Mol Sci* 2012, **2:**242–253.
33. Kresse G, Furthmüller J: **Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set.** *Comp Mat Sci* 1996, **6:**15–50.
34. Salje EKH, Artacho E, Austen KF, Bruin RP, Calleja M, Chappell HF, Chiang G-T, Dove MT, Frame I, Goodwin AL, Kleese-van Dam K, Marmier A, Parker SC, Pruneda JM, Todorov IT, Trachenko K, Tyer RP, Walker AM, White TOH: **eScience for molecular-scale simulations and the eMinerals project.** *Phil Trans R Soc A* 2009, **367:**967–985.
35. Yang XY, Bruin RP, Dove MT: **Developing an end-to-end scientific workflow. A case study using a comprehensive workflow platform in e-science.** *Comput Sci Eng* 2010, **12:**52–61.
36. White TOH, Bruin RP, Chiang G-T, Dove MT, Tyer RP, Walker AM: **Lessons in scientific data interoperability: XML and the *e*Minerals project.** *Phil Trans* 2009, **367:**1041–1046.
37. *FoX library.* http://www1.gly.bris.ac.uk/~walker/FoX/.
38. Murray-Rust P, Hanwell MD, Hutchison GR, Neylon C, Spjuth O, Townsend J, Willighagen E, Walker AM: **Building a CML code library.** *J Cheminfo* 2012, **4:**14.
39. Soler JM, Artacho E, Gale JD, García A, Junquera J, Ordejón P, Sánchez-Portal D: **The SIESTA method for ab initio order-N materials simulation.** *J Phys Condens Matter* 2002, **14:**2745–2779.
40. Gale JD: **GULP - a computer program for the symmetry adapted simulation of solids.** *J Chem Soc Faraday Trans* 1997, **93:**629–637.
41. Glöß A, Brändle MP, Klopper W, Lüthi HP: **The MP2 binding energy of the ethene dimer and its dependence on the auxiliary basis sets: a benchmark study using a newly developed infrastructure for the processing of quantum chemical data.** *Mol Phys* 2012, **110:**2523–2534.

42. Valiev M, Bylaska EJ, Govind N, Kowalski K, Straatsma TP, van Dam HHJ, Wang D, Nieplocha J, Apra E, Windus TL, de Jong WA: **NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations.** *Comput Phys Commun* 2011, **181:**1477–1489.
43. Black GD, Schuchardt KL, Gracio DK, Palmer B: **The Extensible Computational Chemistry Environment: A Problem Solving Environment for High Performance Theoretical Chemistry**. In *Computational Science - ICCS 2003: June 2–4, 2003; Saint Petersburg Russian Federation and Melbourne, Australia.* Edited by Sloot PMA, Abramson D, Bogdanov AV, Dongarra J. Heidelberg: Springer Verlag; 2003:122–131.
44. *CML validator.* http://validator.xml-cml.org/.
45. *eXtensible Data Model and Format (XDMF).* http://www.xdmf.org.
46. Adams N, Cannon E, Murray-Rust P: **ChemAxiom – an ontological framework for chemistry in science.** *Nature Proceedings* 2009. doi:10.1038/npre.2009.3714.1.
47. Guba R, Howard MT, Hutchinson GR, Murray-Rust P, Rzepa H, Steinbeck C, Wegner J, Willighagen EL: **The Blue Obelisk – interoperability in chemical informatics.** *J Chem Inf Model* 2006, **46:**991–998.
48. Schuchardt KL, Didier BT, Elsethagen T, Sun L, Gurumoorthi V, Chase J, Li J, Windus TL: **Basis set exchange: a community database for computational sciences.** *J Chem Inf Model* 2007, **47:**1045–1052.
49. Boyer J: *Canonical XML Version 1.0 W3C recommendation.* 2001. http://www.w3.org/TR/xml-c14n.
50. Clark J, De Rose S: *XML Path Language (XPath) Version 1.0*, W3C recommendation. 1999. http://www.w3.org/TR/xpath/.
51. *FoX wcml documentation.* http://www1.gly.bris.ac.uk/~walker/FoX/DoX/FoX_wcml.html.

doi:10.1186/1758-2946-5-25
Cite this article as: de Jong *et al.*: From data to analysis: linking NWChem and Avogadro with the syntax and semantics of Chemical Markup Language. *Journal of Cheminformatics* 2013 **5**:25.